Fluid Flow on Interacting Deformable Surfaces

Patrick J. Neill*
Oregon State University

Ron Metoyer†
Oregon State University

Eugene Zhang‡
Oregon State University

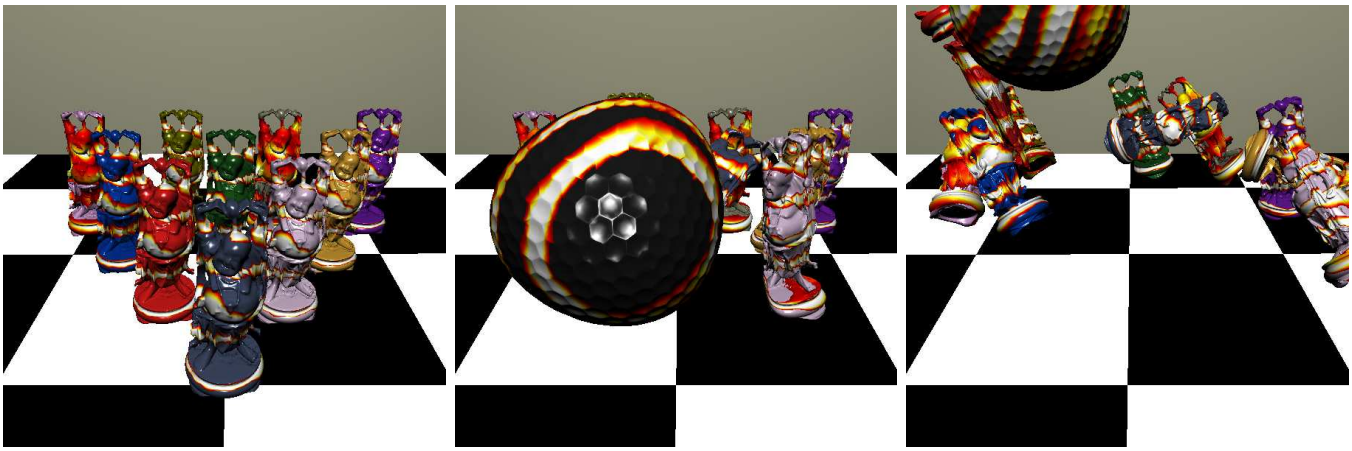


Figure 1: Fluid flow on ten deformable Buddha statues after being hit by a golfball.

Abstract

Fluid simulation on interacting deformable surfaces is a challenging problem that has many applications. In this paper, we present a framework in which artistic as well as physically realistic flows can be generated on surfaces during deformation and collision.

Our simulation system provides comprehensive control over the motion and deformation of an object as well as the movement and density of the fluid on the surface. At the heart of our system is a numerical solver that allows viscous and incompressible flows to be directly generated on surfaces using concepts from differential geometry, such as *geodesic polar maps* and *parallel transport*. This solver is fast and stable even when the object undergoes deformation or collides with other surfaces.

We also propose rules that allow deformation and collisions to impact fluid flows in a physically realistic manner. By combining these rules with a set of comprehensive design functionalities, we develop a system in which the user can specify shape deformation, collision, and fluid flow in a unified framework.

We demonstrate the capability of our system with a number example scenarios.

CR Categories: I.3.7 [COMPUTER GRAPHICS]: Three-Dimensional Graphics and Realism—Animation.

Keywords: fluid simulation, surfaces, deformation, collision, viscosity, parallel transport

*e-mail: neillp@eecs.oregonstate.edu
†e-mail: metoyer@eecs.oregonstate.edu
‡e-mail: zhange@eecs.oregonstate.edu

1 Introduction

Fluid simulation is a complicated problem with a wide variety of applications. Volumetric solutions to simulating realistic fluid flow have addressed a wide range of problems associated with visualizing the complex behavior of fluids, albeit expensively. Fluid flow on deformable surfaces may present an attractive, relatively inexpensive alternative to computing a full volumetric simulation. This is similar to other fields of research in computer graphics, where computations are done in different domains, with different advantages and disadvantages, all with the goal of achieving similar results. Volumetric animation, for example, provides high quality animations and deformation without distortion, much like volumetric fluids. Due to the complexity and size of volumetric animation techniques, alternative mesh-based, or skeletal-based systems are often used due to their excellent work at imitating the more expensive algorithm at a fraction of the cost. Texture parameterization is another case, with surface parameterizations usually being more popular for interactivity when compared to their more computational volumetric counterparts.

It is therefore interesting to realize that little research has been done on the topic of simulating volumetric fluids with surface fluids. This is especially useful since many physically interesting fluid phenomena can be simplified down to a deforming, surface-like representation. Examples of such physical behavior are lava, rain drops hitting water, and other fluids undergoing consistent deformation (i.e. not generating any splashing-like effects). Therefore, it is reasonable to assume that there are a group of volumetric fluid visualizations that are well suited for surface-based approaches.

In addition to having the capability of imitating volumetric fluids in certain circumstances, fluid flow on deformable surfaces also portrays interesting artistic values. Artists, for example, may be able to use the system to interactively design a river-bed by changing and deforming the geometry of a surface in order to get a physically plausible, and desirable river flow. The user may also take advantage of interesting fluid effects by advecting textures across surfaces, synthesizing textures that may have otherwise taken a long

period of time to create.

The set of problems related to the flow of a fluid on deformable surfaces are not easy. One has to first deal with an irregular discretization over which the computational domain is limited to. Multiple, large Partial Differential Equations have to also be solved over the surface in a fraction of a timestep for interactive visualization. Much of the speed up of fluid simulation relies on precomputing most of these quantities. Add deformation into the system, and these quantities, whose constants rely on static geometry become invalid after every timestep. One also has to deal with the ambiguity associated with vectors lying in different tangent planes. Finally, one also has to cope with collisions between deformable objects, a research problem still being actively being studied. Coupling these processes together in a unified system is a topic that has not been discussed before until now.

Therefore, in this paper, we present the following contributions.

1. We present a unified framework and a system for the simulation of fluid flows on deformable and interacting surfaces. To the best of our knowledge, this is the first time such a system is developed.

2. At the core of this system is a numerical fluid solver that not only is stable during surface deformation but also supports efficient computation such as performing velocity diffusion and Helmholtz-Hodge decomposition. As part of this solver, we borrow ideas such as geodesic polar maps and parallel transport from classical differential geometry and apply them on mesh surfaces.

3. We develop rules that allow fluids to be effected by surface deformation in a physically realistic manner, which requires that surface-fluid friction be accounted for.

4. We provide a comprehensive set of functionalities that allows the user to design and control only not the density and velocity of the fluid but also the motion and deformation of the underlying surface.

The remainder of the paper is organized as follows. We first review related work in fluid simulation on surfaces in Section 2 and provide an overview of our simulation system in Section 3. Next, we describe the details of fluid solver in Section 4 and how it is impacted by surface deformation and collision in Section 5. In Section 6, we demonstrate results of our system with a number of scenarios. Finally, we summarize our contributions and discuss some possible future work in Section 7.

2 Related Work

Solving the Navier-Stokes equations has been a thoroughly researched problem. Foster and Metaxas [1997] provide a finite-differencing approach with explicit integration steps to demonstrate gas in a volume. Their use of an explicit integration scheme, however, can incorrectly increase the energy of the system causing the visualization to "blow up" during advection and diffusion for large time steps. Later work published in [Stam 1999] showed that through the use of stable backwards Euler Integration, semi-Lagrangian advection, and implicit diffusion that the previous timestep constraint can be eliminated. Both papers, however, restrict the computational domain to a regular grid of same-sized cells, which makes Helmholtz-Hodge decomposition and semi-Lagrangian tracing easier to define but invalid for irregularities found in triangular meshes. Polthier and Preuss [2000; 2002] define Helmholtz-Hodge decomposition of discrete vector fields on arbitrary surfaces. Their work is lated extended by Tong et al. [Tong et al. 2003] to irregular 3D tetrahedral volumes. Our model of Helmholtz-Hodge Decomposition on surfaces follows the work found in [Tong et al. 2003], but simplifies the tetrahedral representation to triangles.

Work describing fluid flow on surfaces has been less covered than the material mentioned above. The primary problems with fluid flow on arbitrary surfaces is that the semi-Lagrangian advection and velocity diffusion steps are more difficult to define. Stam [2003b] originally proposed the use of Catmull-Clark subdivision surfaces in order transform the Navier-Stokes equations to a parameterized domain where all tangent planes are continuous. This effectively eliminates the ambiguity associated with vectors defined on different tangent planes, allowing tracing and diffusion to occur similarly to the planar case. The use of the Catmull-Clark subdivision surfaces allowed the Navier-Stokes equations to essentially be solved on a surface, but not without noticable artifacts due to the distortion introduced by surface parameterization, and the extra cost incurred by working with such representations. While their technique did much to reduce distortions through the use of a deformation metric, artifacts were still present specifically on the boundaries between patches.

Shi and Yu [2004] presented the idea of solving the Navier-Stokes equations directly on the mesh without the use of special surfaces, and represents the most related work to this paper. The paper used Geodescic Polar maps, contextually found in [Welch and Witkin 1994] and [Polthier and Schmies 1999], to parallel transport vectors defined on a triangles to incident vertices at the center of a one-ring neighborhood. This technique was used in the paper's velocity interpolation scheme after a semi-Lagrangian tracing step. The interpolation scheme combined velocities stored natively at the center of a given triangle with velocities interpolated to the given triangles vertices to ensure piece-wise continuity. Shi and Yu [2004] also presented an interesting advection routine in the paper. For every triangle velocity, a tracing algorithm traversed the mesh, rotating both the intersected triangles and the vectors stored within them onto the same plane as the origin triangle. At the end of this traversal, the aformentioned interpolation scheme was used to acquire, parallel transport, and rotate an interpolated velocity back to the origin. The curved mesh, in this manner, is essentially degenerated into a flat planar representation. In general terms, the advection routine basically wraps the mesh around the advection direction until the tracing has been completed. We feel a more intuitive, per-vertex advection algorithm that makes use of parallel transport to wrap the vector around the mesh provides a significant contribution to the area of research. Shi and Yu [2004] also makes the assumption that the flow is inviscous.

3 Overview

The pipeline of our system contains four components (see Figure 2). Each part of the simulation can be operated independently, allowing the user to operate interactive on whatever part of the project they are currently focused on.

1. **Fluid simulation** is responsible for generating stable fluids on a surface regardless of changes in curvature and volume. The field is subject to forces introduced by the user control module, or by physical forces such as gravity, surface-fluid friction, and other various forces passed on by the collision detection and response module. The fluid behavior also changes based on how the surface deforms. Because of the recomputation of the constants encoding geometric information, fluid may be stretched, flow quicker, or change course during large deformation. Failure to recompute these constants will create

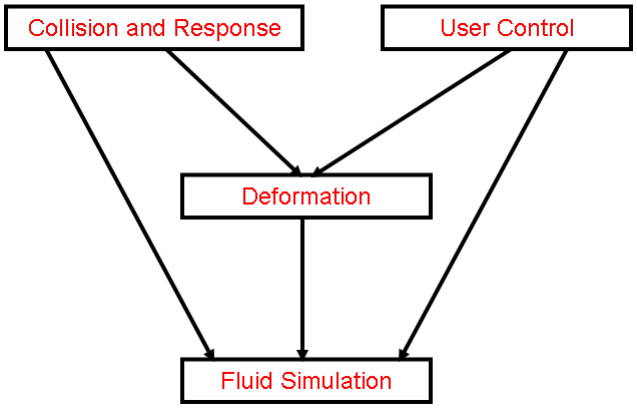


Figure 2: This figure illustrates the pipeline of our system.

large artifacts throughout the simulation, as local coordinate frames and weighting schemes quickly become invalid under even small deformations.

2. **Deformation simulation** allows the motion and shape of the underlying surface to be modified either by the user or through interactions with other objects in the scene. Deformations are applied to the model in three different situations. The first type of deformation responds to user controlled pulls by attaching a spring between the point of contact and the user's mouse. The second type of deformation occurs during collisions. The deformation system takes force information from the collision detection and response system and generates physically plausible deformations based on the current context. The third type of deformation deforms the surface based on underlying density values. By using density sources essentially as displacement scalars, we perturb the geometry to give it a user-guiding look of flowing lava, simulating the concept of volumetric fluids on surfaces. Deformation occurs after the physics module has completed. After this step, the Sphere Bounding Heirarchy may also need to be recomputed if the model displacement is too large.

3. **Collision and Response** is responsible for generating forces based on collisions and global effectors such as gravity or wind for a given timestep. Runge-Kutta 4 is used to perform explict integration over a timestep. During collisions, impulse forces are calculated similar to [Baraff 2001] and are also passed on to the deformation system. Such forces are introduced to the deformation and fluid module in order to generate the effects present in this paper. A Dynamic Sphere Bounding Hierarchy [Quinlan 1994] is used to accelerate intersection tests, rebuilding itself after each large deformation. Certain types of forces, such as the change in angular and linear velocity denoting acceleration, are also introduced into the fluid simulator.

4. **User Control** The user of this tool has a variety of functionality in order to control the fluid flow. First, the velocity field of the fluid can be turned on/off, giving the artist interactive control over the simulation for models previously restricted to offline manipulation. Density values can be halted, placed, and diffused at different rates based on user controlled parameters. Deformation can be increased/decreased dynamically, giving the user control over how their objects interact with the environment. A user-controlled dampening force is also applied to increase/decrease the time it takes for the model to return to it's rest. This allows slow, heavy movements to take place alongside quick, spastic deformations. In short, the user controls the fluid/surface through deformation forces, and explicit fluid forces.

We will describe each module in detail in the next sections.

4 Solving Navier-Stokes Equations on Surfaces

In this paper, we focus on simulating viscous incompressible flows on interacting deformable surfaces. Such flows can be described by the following form of the Navier-Stokes equations:

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u}\cdot\nabla)\mathbf{u} - \frac{1}{\rho}\nabla p + \nu\nabla^2\mathbf{u} + \mathbf{f} \quad (1)$$

$$\nabla\cdot\mathbf{u} = 0 \quad (2)$$

where $\mathbf{u}$ is the flow velocity and and $\mathbf{f}$ is the external forces that are applied to the fluid. Both are vector fields. ρ represents the density of the fluid, p is the pressure field, and ν is the kinematic viscosity. These equations are direct consequences of conservation of momentum and mass.

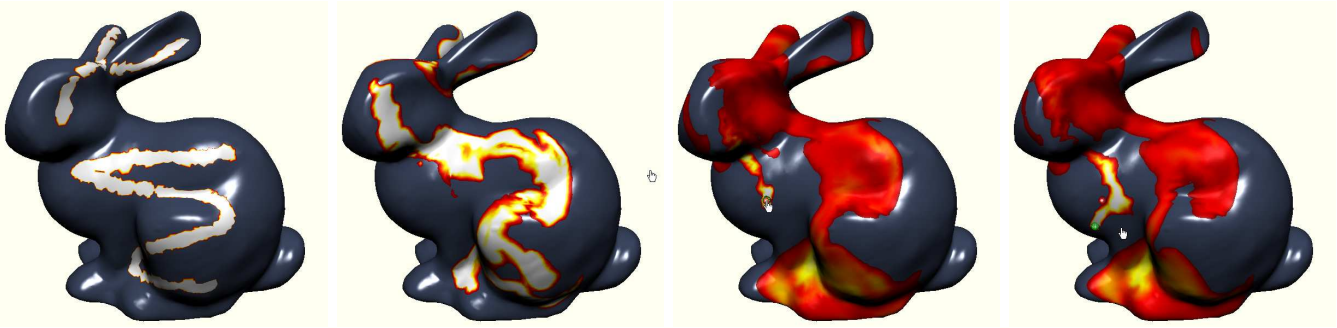
While there have been many excellent techniques for solving the Navier-Stokes equations in 3D volumes, few solvers exist for curved surfaces. To the best of our knowledge, there has been no prior work on solving these equations on deformable surfaces that are subject to collisions.

Our fluid solver uses the following computational setup. First, the underlying manifold surface is represented by a triangular mesh, with vertices $V = \{v_i | i = 1,...l\}$, edges $E = \{e_j | j = 1,...m\}$, and triangles $T = \{t_k | k = 1,...n\}$. All the variables in the above equations are maintained at the vertices. Piecewise interpolation schemes are used to obtain values on the edges and inside triangles. For scalar values such as fluid density ρ, pressure p, and the kinematic viscosity ν, we use the barycentric interpolation scheme that leads to a linear interpolation. For vector fields such as fluid velocity $\mathbf{u}$ and external force $\mathbf{f}$, it has been shown that piecewise linear combinations of the vector values at the vertices do not guarantee continuous vector fields after the interpolation, and we therefore use the non-linear interpolation scheme of [Zhang et al. 2006] which guarantees continuity after interpolation. However, a vector-based interpolation scheme ensures a smaller support for interpolation (one triangle) than a triangle-based scheme (at least four triangles). Otherwise, our framework resembles that of Shi and Yu [2004]. During every timestep t, we first obtain the new velocity $\mathbf{u}^{t+1}$ and then the fluid density ρ^{t+1}.

To compute $\mathbf{u}^{t+1}$, we perform the following operations in this order.

1. $\mathbf{u}_d$ = diffuse($\mathbf{u}^t$)
2. $\mathbf{u}_a$ = advect($\mathbf{u}_d$)
3. $\mathbf{u}_f = \mathbf{u}_a + \mathbf{f}^t$
4. $\mathbf{u}^{t+1}$ = project($\mathbf{u}_f$)

Adding external forces (Step 3) is straightforward. Later, we will describe how to generate external forces based on shape deformation and collision. To perform advection and diffusion surfaces, we must overcome the difficulty that a surface in general lacks a global parameterization. There have been techniques building local parameterizations by diving the surface into a number of



(a) (b) (c) (d)

Figure 3: Fluid flow on the motionless rigid Bunny surface. Notice the user can interactively add density to the fluid (in (c), the fluid is being added near the mouse location).

patches [Sander et al. 2001; Lévy et al. 2002; Desbrun et al. 2002; Sorkine et al. 2002; Zhang et al. 2005], each of which is a collection of triangles that form topological disks. Distortion in the parameterization can lead to incorrect results in computing diffusion unless it is accounted for, and seams between patch boundaries can lead to errors during advection. Our solution is to solve the Navier-Stokes equations on the surface directly. This can be seen as a parameterization in which every triangle is a patch. Unfolding a single triangle results in no distortion, and we use the interpolation scheme of Zhang et al. [2006] that ensure vector field continuity across edges, which are the seam boundaries.

This requires the ability to define the Laplacian at a vertex based on vector values at incident vertices, which can be solved by using the idea of parallel transport. We later use the same idea to transfer vectors between the vertex-based representation and the triangle-based representation during the projection state. Therefore, we first describe the idea of parallel transport on smooth surfaces and on meshes, which is linked to the concept of geodesics.

A geodesic on a curved surface is a locally shortest and straightest curve. It is a generalization of a straight line in the plane. Given a surface S and two points $\mathbf{p},\mathbf{q} \in S$, there is a *geodesic* γ connecting them, i.e., $\gamma(0) = \mathbf{p}$ and $\gamma(1) = \mathbf{q}$. Let $V_\mathbf{p}$ and $V_\mathbf{q}$ be tangent vectors defined at p and q, respectively. If the oriented angle between $\gamma'(0)$ and $V_\mathbf{p}$ equals that between $\gamma'(1)$ and $V_\mathbf{q}$, then $V_\mathbf{p}$ and $V_\mathbf{q}$ are *parallel* with respect to γ , and $V_\mathbf{q}$ is said to be the *parallel transport* of $V_\mathbf{p}$ along γ. γ gives rise to an orthonormal and bijective linear map between $TM_\mathbf{p}$ and $TM_\mathbf{q}$, the tangent planes at $\mathbf{p}$ and $\mathbf{q}$.

To solve the diffusion component, we make use of the following equations:

$$\mathbf{u}_i = \sum_{j \in J} \omega_{ij} T_{ij}(\mathbf{u}_j) \tag{3}$$

where $\mathbf{u}_i$ is the vector value at vertex v_i, ω_{ij} is the mean value coordinate of Floater's [2003], and T_{ij} is the transport function from the tangent plane at vertex v_j to vertex v_i. Let $\begin{pmatrix} F_i & G_i \end{pmatrix}^\mathbf{T}$ be the coordinates of $\mathbf{u}_i$ under the local frame at vertex v_i. Equation 3 has the following more explicit form:

$$\begin{pmatrix} F_i \\ G_i \end{pmatrix} = \sum_{j \in J} \begin{pmatrix} \cos\Delta\theta & -\sin\Delta\theta \\ \sin\Delta\theta & \cos\Delta\theta \end{pmatrix} \begin{pmatrix} F_j \\ G_j \end{pmatrix} \tag{4}$$

where $\Delta\theta$ is the difference between the angle from the X-axis to the geodesic at $\mathbf{p}_2$ to that from X-axis to the the geodesic at $\mathbf{p}_1$.

To perform the Hodge decomposition, we compute a pressure field p by solving the discrete equations $\text{div}\nabla h = \text{div}\,\mathbf{u}$. Our implementation follows closely of that of [Polthier and Preuss 2000], which requires a triangle-based vector field representation. This requires the ability to transfer $\mathbf{u}$ between the vertex- and triangle-based representations. Consider the case of vertex-to-triangle transition. Given a triangle $t = \{v_1, v_2, v_3\}$ and vector values $\mathbf{u}_{i,t}$ at v_i, the vector value at the barycenter of t is

$$\mathbf{u}_t = \sum_{i=1}^{3} T_{i,t}(\mathbf{u}_{i,t}) \tag{5}$$

where $T_{i,t}$ is the transfer function that maps a tangential vector U at v_i to a tangent vector in T. Typically, this is done by casting U as a 3D vector and project it onto the plane containing T. Notice this approach leads to large errors in high-curvature regions of the surface since the magnitude of U will not be maintained. We define $T_{i,t}$ to the parallel transfer function from v_i to the barycenter of t along the geodesic connecting them, which is a straight line segment. Recall $T_{i,t}$ introduces a bijective map between the tangent plane at v_i and T and it maintains the magnitude of tangent vectors. These properties make it a more reliable transfer function.

After the Hodge decomposition, we map the vectors from the triangles back to the vertices again through parallel transport.

With $\mathbf{u}$, we can now solve for the density by first adding new density, then advecting it along $\mathbf{u}$ in a similar fashion as the advecting a vector field, and finally perform scalar-valued Laplacian smoothing.

In the next section, we will introduce external forces due to surface motion, deformation, and collision. It will become clear that such forces are not always divergence-free. This requires us to make modifications to our process in order not to lose the effect of such forces.

5 Impacts of Deformation and Collision on Flow

In this section, we describe how the fluid is impacted by the motion and deformation of the underlying surface. While our approach is physically based, we also have to account for the restriction that the fluid cannot leave the surface of the object. This leads to a number of rules that we describe next.

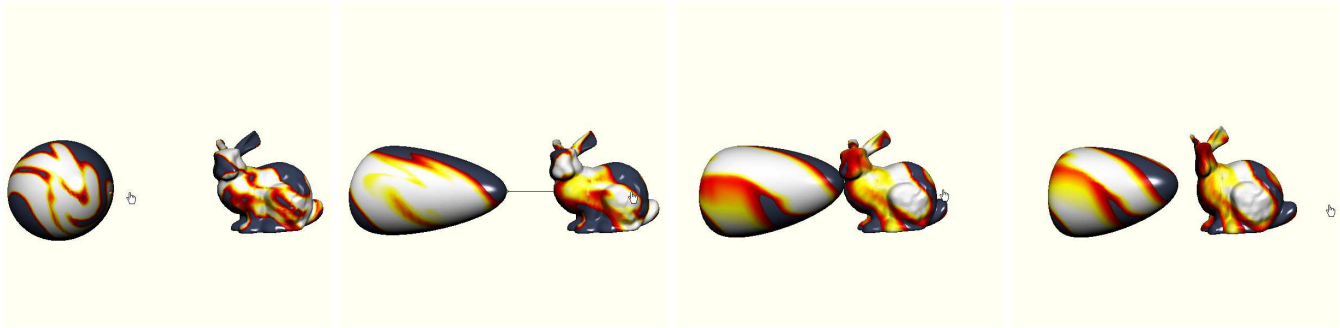


Figure 4: This figure demonstrates our capability to allow a user to design deformation and to guide inter-object interaction. Furthermore, notice our fluid solver produces realistic flows during shape deformation and collision.

5.1 Deformation Design

The deformation algorithm used by our system in our system is meshless deformation based on shape matching first presented in [Muller et al. 2005]. In typical mass-spring models with explicit integration schemes, like forward Euler or Runge-Kutta, care must be taken in chosing the timestep and spring stiffness due to the capability of overshooting the equilibrium. This constraint is explained more thoroughly in [Muller et al. 2005], but is commonly found in most explicit integration schemes, including fluid dynamics and collision detection/response where spacetime discretization is used.

Meshless shape matching addresses the problem of updating velocity by creating an unconditionally stable integration scheme which does not overshoot the equilibrium, thus conserving momentum. In this scheme, the mesh is represented by two sets of particles, the mesh in its transformed deformed state, and the mesh in its resting static state. By matching up a particle in the deformed position with the static counterpart, the algorithm can design a integration scheme which moves the particle towards a physically plausible goal position without ever erroneously increasing the energy of the system.

The first step in this algorithm is to find the transformation matrix which matches the particles representing the static mesh to the deformed state. Let x_i^0 and x_i be the two sets of particles of mass m_i that represent the static and deformed shapes of the mesh respectively. The goal is to find a rotation matrix **R**, and translation vector **t** which best describes the transformation of the static mesh to its deformed state. The translation vector, **t**, is simply the difference in the center of mass of the static state, with the center of mass of the deformed state, i.e. the vector that translates the center of the static mesh to the center of the deformed one. Computing the rotation matrix **R** involves first computing a matrix **A** with the following conditions, as described in [Muller et al. 2005]:

$$q_i = x_i^0 - x_{cm}^0$$
$$p_i = x_i - x_{cm}$$
$$A = (\sum_i m_i p_i q_i^T)(\sum_i m_i q_i q_i^T)^{-1} = A_{pq} A_{qq}$$

The rotational part **R** is found by Polar Decomposition, [Shoemake and Duff 1992], with $R = A_{pq} S^{-1}$ where $S = \sqrt{A_{pq}^T A_{pq}}$ represents the symmetric, non-deforming part of the transformation. Evaluating the square root of a function is done by first diagonalizing the matrix through a method like Jacobi rotation. After this is done, the square root is then applied to the eigenvalues of the system before recompositing it back in its computed form.

Once **R** has been found, the goal positions and the subsequent updates to particle velocities are found by rotating the original particle, $x_i^0 - x_{cm}^0$, by **R**, then adding x_{cm}. By linearly interpolating **A** with the rotation matrix **R** during the transformation process, stretching and shearing can be introduced into the shape matching. Following the same idea, twisting and bending can be created by computing a larger matrix that represents a quadratic match, as described in [Muller et al. 2005], and performing the linear interpolating with it, instead of **A**.

The interactivity of this approach made it a good choice for the system we developed in this paper. Currently, the Meshless Deformation applies the computed deformation globally across the whole model, making it less suitable for objects with complex, varying geometry undergoing large deformations. The original authors solved this problem through overlapping clusters subdivided across the model. These clustered are then matched to their original configurations, and the particles contained within are updated appropriately. We are currently looking into using the Sphere Bounding Heirarchy tree used by our physics system to spatially partition the model into overlapping regions. Different levels of the tree will demonstrate different levels of stiffness, increasing or decreasing the amount of clusters matched based on dynamically changing stiffness or user interaction.

5.2 Collision Detection

To provide interaction between multiple deformable surfaces, we have implemented a standard collision detection system with a hybrid collision response system that imparts forces onto the object as a rigid body to produce rigid body motion and imparts motion to the vertices to produce deformations.

The linear and angular motion of each deformable object is governed by rigid body dynamics. As objects move through space, we use a hierarchical bounding sphere algorithm to efficiently identify collisions with other deformable surfaces [Quinlan 1994]. Because our objects deform over time, the bounding spheres are updated at each frame.

Our system runs at a time step determined by the user, which is typically set to 1/60 to 1/30 seconds for real-time interaction. We use a Runge Kutta (RK4) integration scheme to integrate the rigid body dynamics forward in time. The RK4 timestep is further subdivided to provide a reasonable timestep for detecting and responding to collisions without unacceptable penetration. Collisions are detected using simple vertex-plane computations. Once a collision is successfully detected, we employ an impulse response method to compute and apply forces directly into the rigid body dynamics [Baraff 2001]. The impulse force is also used to scale the position

Figure 5: Example of physical forces introduced by the decoupling of the surface velocity and the fluid velocity. Acceleration of the surface causes the fluid to become more agitated as frictional forces attempt to keep the fluid stationary.

of the colliding vertex, resulting in a deformation that propagates through the surface. We currently do not support resting contact.

5.3 Deformation-introduced Forces

To understand and emulate the impact on the fluid by the motion and deformation of the underlying surface, we consider the following scenario. A passenger is on a train that is constantly changing directions and magnitudes. The passenger will feel a force that is pushing him in the opposite direction of the changes in the train's velocity. We refer to this force as the *inertia force*.

Consider a surface S with the mass M, velocity $v(t)$, and external force $F(t)$. The instant acceleration is $\frac{F(t)}{M}$. Given a small concentration of fluid in S with a mass m, the inertia force by S on the fluid is then

$$f_i(t) = -m\frac{F(t)}{M}. \tag{6}$$

The inertia force is used to account for the impact on the fluid by the changes in motion and shape of the underlying surface.

We also consider the friction between the surface and fluid, which is $f_r = \alpha v$ where v is the fluid velocity relatively to the underlying surface. Then the total force that exercised by the surface to the fluid is $f_i + f_r$.

Essentially what this states is that the fluid in this representation is seperate from the ground, and is only bound to it by frictional forces. If the floor moves out from under it, a fluid with $\alpha = 1$ stays stationary with respect to the surface point under it, while a fluid with $\alpha = 0$ moves completely independent of the surface it lies on. This can violate the divergence-free constraint of the Navier-Stokes equation in the case of linear acceleration, as Δv projected onto a spherical objects Mesh essentially constitutes a source-sink.

Collisions also introduce locally ill-suited vector fields at their point of contact, signifying possible compression of the fluid similar to exposive shockwaves. Angular acceleration, however, is well suited, resolving in the case of the sphere to a pure rotational field. Further exploration into simulating compression would be needed to refine these rules, and to ensure mass is conservated during compressibility.

6 Results

The combination of physics, fluid flow on surfaces, and collision response provides us with interesting visualization opportunities. The tool developed for this paper allows the user to toggle different parts of the simulation, allowing interactive placement of forces, fluid, and physics based collisions, even if the full simulation runs at less than interactive speeds.

Figure 3 demonstrates fluid flow on static surfaces. This was done at interactive rates, where the user can place and push density sources around the surface for, like in the bunny's case, meshes of even 40k triangles. Viscosity is also a user defined parameter, where different values represent the speed density values or velocity vectors diffuse across the surface.

Figure 4 demonstrates stable fluid simulation on a surface undergoing heavy deformation. The deformation in this scenario causes constants previously computed off of the weights to become invalid. This values had to be quickly recomputed, in addition to an expensive rebuild of the Sphere Bounding Hierarchy. Even with these constraints in place, our system achieves interactive results for fluid flow on deformable, colliding surfaces.

Figure 5 shows the effect our physics simulation has on the property of the fluid. By taking into account forces generated by the change in velocity into the behavior of the fluid, interesting visual effects are shown. At the beginning of the simulation at timestep t_0, the fluid's velocity with respect to the surface is zero. Once the object moves, however, the change in the objects velocity causes the fluid to "lag" behind on the surface until it has the capability to catch up. This is simulated through the introduction of physical forces (see Section 5).

The scene shown in Figure 6 demonstrates the capabilities of the system we have developed. Ten happy buddha models consisting of 60k triangles are arranged with different density and fluid viscosity terms, along with different deformation stiffness. A golfball consisting of 50k triangles is propelling towards the buddha formation by the use of a spring with a high spring constant. Each frame took approximately 25 seconds to render on a Pentium IV 3.8Ghz processor. The scene represents our systems ability to handle large data sets, with multiple collisions, deformations, and fluid flow occuring across the entire scene.

Figure 7 shows another capability fluid on surfaces has when the surface is allowed to be deformable. By using the density value as a displacement map, fluid flow can be used to interactively deform the model into interesting, and artistically difficult to create shapes. When incorporating the newly displaced geometry into the fluid simulation, fluid will begin to flow differently across the newly deformed shape, demonstrating a coupling between deformation and fluid flow. Melting, or disintegrating effects can be easily and interactively generated by using the above approach. This newly deformed geometry also interacts physically with the environment in a similar fashion as the regular surface based visualization. Coupling this effect with a more advanced description of physics can be thought of as the first major step in simulating volumetric fluids with surfaces.

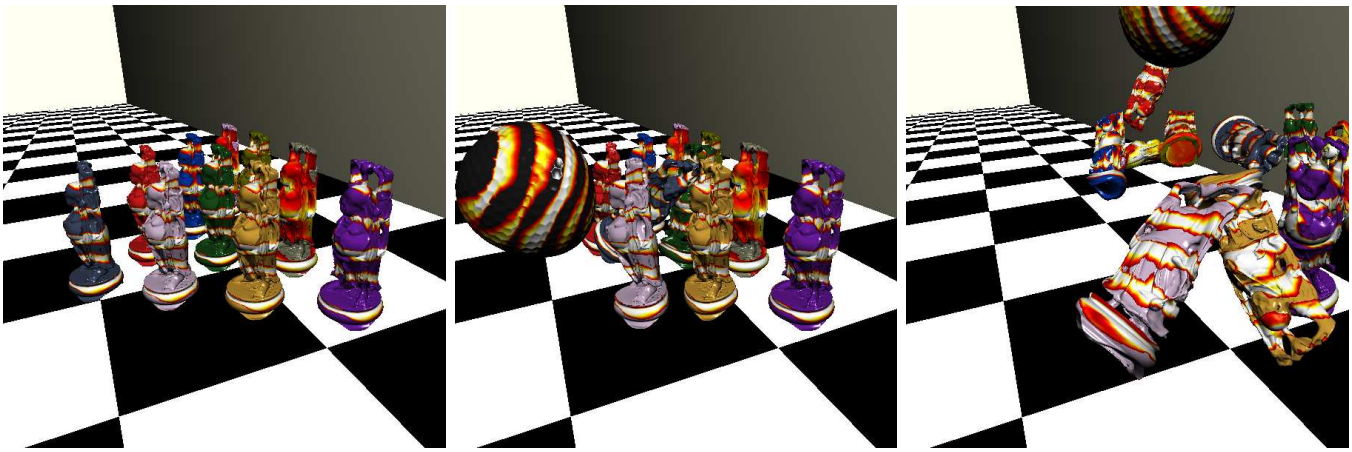


Figure 6: A large scene consisting of 650k+ Triangles, with deformations, collisions, and viscous fluid flow.

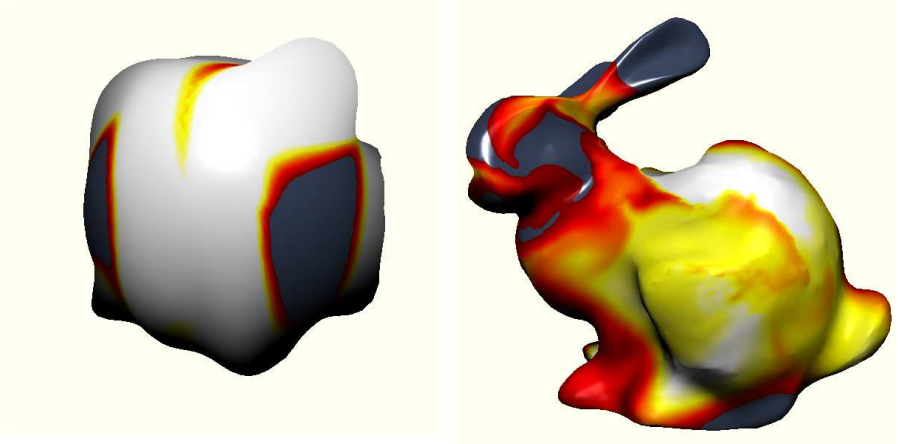


Figure 7: Two effects achievable by our system: a vertex shader used to displace geometry based on density scale (left), and an interactive lava shader dissolving geometry (right).

7 Conclusion

In this paper, we have developed a system to simulate fluid flow on deformable surfaces that can undergo collision. To our knowledge, this is the first time such a system is developed.

Central to our system is a numerical solver that supports fast and stable simulation of fluid flows on surfaces during shape motion, deformation, and collision. The solver relies on a vertex-based vector field representation and the ability to perform parallel transport between a vertex and another vertex or triangle, which is based the concepts of geodesic polar maps and parallel transport.

We describe a set of rules that allows the fluid to be effected by the motion and deformation in a physically realistic manner. These rules are based on the concepts of inertia and friction between the fluid and the underlying surface.

Our system also provides the user with a wide range of control capabilities such as the velocity and density of the fluid and the motion and deformation of the surfaces.

There are a number of places in our system that we wish to improve upon.

First, our solver does not account for the impact on fluid by inter-object collision. This is mainly due to the paradox introduced by the Helmholtz-Hodge decomposition. On one hand, collision forces tend to be curl-free. When it is applied before the Helmholtz-Hodge decomposition, its impact will be often lost. On the other hand, when the force is applied to the density function of the fluid directly, we typically see a high concentration of fluid which will disappear. This is a clear violation of the mass conservation principle. We plan to further investigate the issue. The work of Yngve et al. [Yngve et al. 2000] is promising due the compressible flows that they handle in the initial stage of explosion simulation.

Second, the largest bottleneck of our current system is the Helmholtz-Hodge decomposition on surfaces. This is primarily because of the need to interpolate values to and from per-vertex/per-triangle representation. The projection steps computational domain runs per-triangle, which makes it more expensive to setup and compute. A vertex-based decomposition algorithm would vastly improve frame rate, but little research has been done in this area. For running the simulation on meshes consisting of 100K+ vertices, a dynamic sampling algorithm in addition to an interpolation step could alleviate most of the computational bottleneck. We have also recognized the achievements done by Elcott et al. [Elcott et al. 2007] as a way to improve the quality of the fluid simulation, and are looking into it for future revisions.

Third, our system does not perform remeshing, which may be useful when a model undergoes permanent, user or physically induced, deformation.

Another future direction is investigate the use of our system for performing texture synthesis on deformable surfaces.

References

ALLIEZ, P., COHEN-STEINER, D., YVINEC, M., AND DESBRUN, M. 2005. Variational tetrahedral meshing. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, ACM Press, New York, NY, USA, 617–625.

BARAFF, D., 2001. Rigid body dynamics.

CARLSON, M., MUCHA, P. J., AND TURK, G. 2004. Rigid fluid: animating the interplay between rigid bodies and fluid. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, ACM Press, New York, NY, USA, 377–384.

DESBRUN, M., MEYER, M., AND ALLIEZ, P. 2002. Intrinsic parameterizations of surface meshes. In *Proceeding of Eurographics*, 209–218.

ELCOTT, S., TONG, Y., KANSO, E., SCHRöDER, P., AND DESBRUN, M. 2007. Stable, circulation-preserving, simplicial fluids. *ACM Transactions on Graphics 26*, 1.

FLOATER, M. S. 2003. Mean value coordinates. *CAGD*, 20, 19–27.

FOSTER, N., AND METAXAS, D. 1997. Modeling the motion of a hot, turbulent gas. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 181–188.

JAMES, D. L., AND PAI, D. K. 2004. BD-Tree: Output-sensitive collision detection for reduced deformable models. *ACM Transactions on Graphics (SIGGRAPH 2004) 23*, 3 (Aug.).

KLINGNER, B. M., FELDMAN, B. E., CHENTANEZ, N., AND O'BRIEN, J. F. 2006. Fluid animation with dynamic meshes. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, ACM Press, New York, NY, USA, 820–825.

LÉVY, B., PETITJEAN, S., RAY, N., AND MAILLOT, J. 2002. Least squares conformal maps for automatic texture atlas generation. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, 362–371.

MULLER, M., HEIDELBERGER, B., TESCHNER, M., AND GROSS, M. 2005. Meshless deformations based on shape matching. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, ACM Press, New York, NY, USA, 471–478.

POLTHIER, K., AND PREUSS, E., 2000. Variational approach to vector field decomposition.

POLTHIER, K., AND PREUSS, E., 2002. Identifying vector fields singularities using a discrete hodge decomposition.

POLTHIER, K., AND SCHMIES, M. 1999. Geodesic flow on polyhedral surfaces. In *Data Visualization '99*, E. Gröller, H. Löffelmann, and W. Ribarsky, Eds. Springer-Verlag Wien, 179–188.

QUINLAN, S. 1994. Efficient distance computation between non-convex objects. *IEEE International Conference on Robotics and Automation*, 3324–3329.

SANDER, P. V., SNYDER, J., GORTLER, S. J., AND HOPPE, H. 2001. Texture mapping progressive meshes. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, 409–416.

SHI, L., AND YU, Y. 2004. Inviscid and incompressible fluid simulation on triangle meshes: Research articles. *Comput. Animat. Virtual Worlds 15*, 3-4, 173–181.

SHOEMAKE, K., AND DUFF, T. 1992. Matrix animation and polar decomposition. 258–264.

SORKINE, O., COHEN-OR, D., GOLDENTHAL, R., AND LISCHINSKI, D. 2002. Bounded-distortion piecewise mesh parameterization. In *VIS '02: Proceedings of the conference on Visualization '02*, 355–362.

STAM, J. 1999. Stable fluids. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 121–128.

STAM, J., 2003. Real-time fluid dynamics for games.

STAM, J. 2003. Flows on surfaces of arbitrary topology. *ACM Trans. Graph. 22*, 3, 724–731.

TESCHNER, M., KIMMERLE, S., ZACHMANN, G., HEIDELBERGER, B., RAGHUPATHI, L., FUHRMANN, A., CANI, M.-P., FAURE, F., MAGNETAT-THALMANN, N., AND STRASSER, W. 2004. Collision detection for deformable objects. In *Eurographics State-of-the-Art Report (EG-STAR)*, Eurographics Association, Eurographics Association, 119–139.

TONG, Y., LOMBEYDA, S., HIRANI, A. N., AND DESBRUN, M. 2003. Discrete multiscale vector field decomposition. *ACM Trans. Graph. 22*, 3, 445–452.

VAN WIJK, J. Image based flow visualization for curved surfaces.

WELCH, W., AND WITKIN, A. 1994. Free-form shape design using triangulated surfaces. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 247–256.

YNGVE, G. D., O'BRIEN, J. F., AND HODGINS, J. K. 2000. Animating explosions. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, 29–36.

ZHANG, E., MISCHAIKOW, K., AND TURK, G. 2005. Feature-based surface parameterization and texture mapping. *ACM Transactions on Graphics 24*, 1, 1–27.

ZHANG, E., MISCHAIKOW, K., AND TURK, G. 2006. Vector field design on surfaces. *ACM Trans. Graph. 25*, 4, 1294–1326.